Docket No. RM01-5-000
Electronic Tariff Filings

Word-To-eTariff Extraction Application

The NOPR in Docket No. RM01-5-000 proposes that all FERC regulated tariffs must be converted to a new electronic format by a date certain. The current draft eTariff software provides for two methods to take existing tariffs in an electronic format and converting them to the proposed electronic tariff format.

First, the tariff can be cut from the native format and pasted into the eTariff format on a section-by-section basis. This feature is most efficient for tariffs up to 300 or 400 pages.
Second, eTariff can import a tariff that has been put into a structured format along with a database explaining that format. That feature is available from the eTariff menu bar at <File><MS Word Interface Import>. This feature requires the writing of a software and organizational format specific extraction program, and is best reserved for longer tariff conversion projects.

This document explains eTariff's "MS Word Interface Import" feature and how to take advantage of this feature to speed-up the tariff conversion process by writing extraction applications for software and organizational format specific tariff documents.

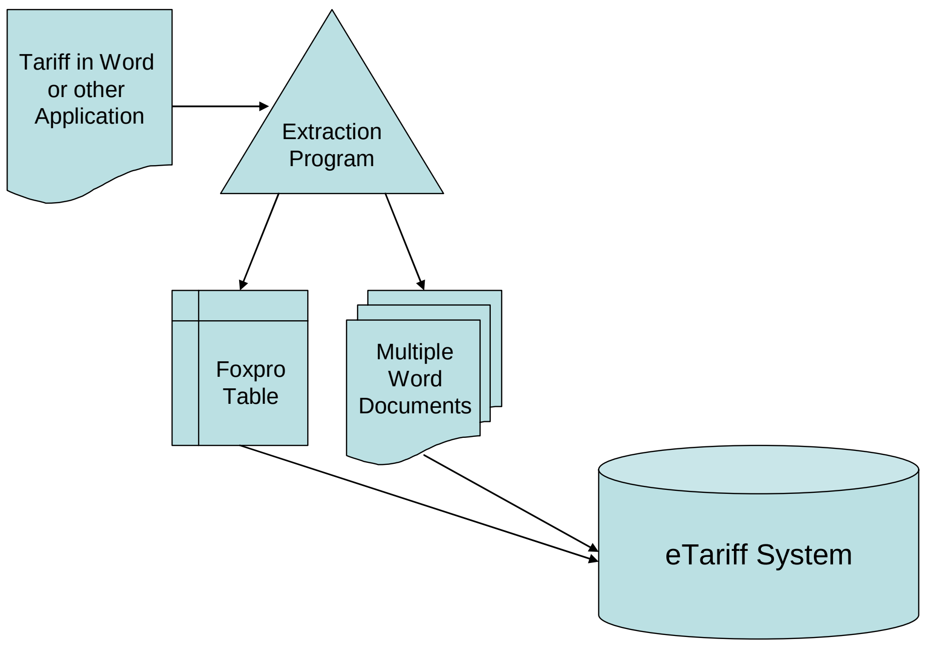
System Overview

The object of the automated import feature is to speed-up the conversion of tariffs in an electronic format to the new eTariff format. This eTariff feature requires

a) A tariff in an electronic format that is subject to standard cut and paste features. This is supplied by the regulated company.
b) An extraction application specifically written for the software format that the electronic version of the tariff is in, and for the organizational format of the tariff document. This is written or supplied by the regulated company.
c) A structured Foxpro database and associated Word files created by the extraction application in the format required by eTariff. This structure is described below.

A diagram of the import process is as follow:

eTariff Import Flowchart
System Overview



Import Interface Specification

The import data consists of 2 types of files:
- MS Word format files, 1 for each ‘section’ of a tariff
- a ‘definition’ file, DBF format, containing data about each MS Word file (above)

All these files must reside in the same directory (folder) when the import process is invoked.

The definition file (table) contains 1 record (row) for each MS word file. Each row in the definition file holds information regarding a single section of the tariff. The format of the definition file is as follows:

Field name	**Data Type**	**Description**
assoc_file	C(80)	The name of the associated MS Word file (aka, the file containing a single section)
sect_id	N(6)	A unique number assigned by the program, used to uniquely identify a section
parent_id	N(6)	The sect_id (above) which the current section belongs to. For ‘top level’ sections this number should be 0
sect_num	C(50)	The tariff section number. This is not to be confused with the section ID. This number may be of the x.x.xx.xxx format.
sect_title	C(200)	The title of the tariff section. Do not include the tariff section ‘number’ as part of the title.

A database file is necessary to guide the eTariff importation software. This database file must of FoxPro 7.x or greater format. The extraction program requires an ODBC/ADO driver for FoxPro to interface with the database file. This driver is free and can be downloaded from Microsoft at: http://msdn.microsoft.com/vfoxpro/downloads/odbc.asp. Information on this driver is available at: http://support.microsoft.com/?kbid=277772

Import Process

The import process is invoked in the eTariff software through the <File><MS Word Interface Import> command.

1) the user selects the “MS Word Interface Import”
2) the user enters the Tariff name[1]
3) the user selects the directory (folder) and the .DBF file
4) the system performs the verification checks mentioned above and report accordingly
5) the system steps through the ‘definition’ file, adding records into the new tariff based

[1] The point of contact information may be entered for this tariff at this time. But it is not necessary. OMTR recommends that users treat this “named” tariff as a working document. Once final edits are made, then either copy the finished tariff into a newly named tariff or rename the tariff in preparation for filing with the Commission.

on information in the ‘definition’ file
6) the system generates additional needed data during the import process (e.g. internally generated data items)
7) the system displays the appropriate response/report as a result of the import process

During the import process, the following will be checked to verify the import format:

1) the field names in the definition file
2) the fields are of the correct data type
3) all fields for each record must have values
4) no duplicate sect_id values
5) every parent_id value exists (or is 0 – zero) in the provided sect_id values
6) every file listed in the assoc_file field exists (in same directory as the definition file)

If any of these criteria are not met, the import process will be halted. A brief message/report will inform the user which record(s) caused the problem.

Extraction Applications

Each company’s tariff is maintained using various office applications and will have a unique organizational formats. For this reason a custom extraction application will be required for each document’s software and organizational format.

FERC will not provide or write these extraction programs. There are too many different software formats and every tariff document has a unique organizational structure. However, OMTR has created some examples that companies’ IT departments may wish to adapt to their electronic software and tariff organizational formats.

Example Extraction Application Detailed Design

An extraction application must perform two major functions:

(1) Parse a tariff by section, create a Word file for that section, and paste, name and save the Word file;
(2) Insert the relevant section and Word File information in a Foxpro database readable by the eTariff software.

The parsing function is the most difficult part of the extraction application to write. Tariffs come in a wide range of organizational formats (I.A.1.a.i, 1.1.1.1, etc.). Further, many older tariffs contain different organizational formats within the same document. In addition, style and font formats can vary. To reduce the coding effort, OMTR recommends that, as with the case of hand-coping and pasting into eTariff, first standardize formatting and styles. Then focus on what feature of the tariff is repeatable and is a predictable indication of a “section.” Edit accordingly to introduce those features into areas that that deserve to be a section.

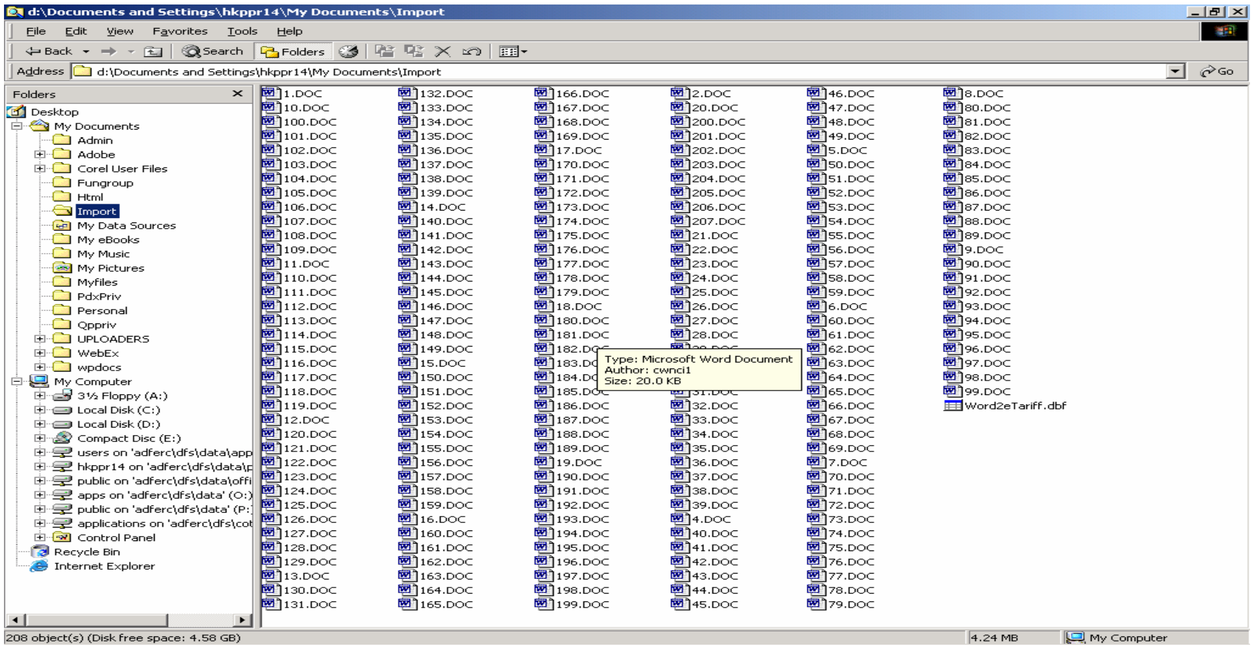
With an edited tariff, the parsing function of the extraction application can be written.

On the Appendix are OMTR's VBA codes for parsing and creating the database for importing the parsed tariff into eTariff. They were written by Casey Nutsch, OMTR, mailto:casey.nutsch@ferc.gov. They can be used as model for other Word documents and other software formats such as Excel. Users of any extraction application should not expect 100% accurate results. The more reasonable objective is to get most of the tariff into eTariff, and then clean it up using the text and section-level editing tools available in eTariff.

Neither OMTR nor the Commission will support these macros. They are only presented to provide guidance for those who wish to automate the tariff conversion process. If these macros are used, neither OMTR, the Commission or its staff takes any responsibility for its use, application or results.

The examples are not the only way to create a database for importing a parsed tariff into the eTariff software. And we are sure many innovations and technical improvements could be made to the example we have created. Further, OMTR's experience is that any tariff up to 300 to 400 pages can be converted by hand in less time than it takes to write and error-check an extraction application.

Running one of the example extraction programs on a large tariff file results in a set of documents such as those shown below:

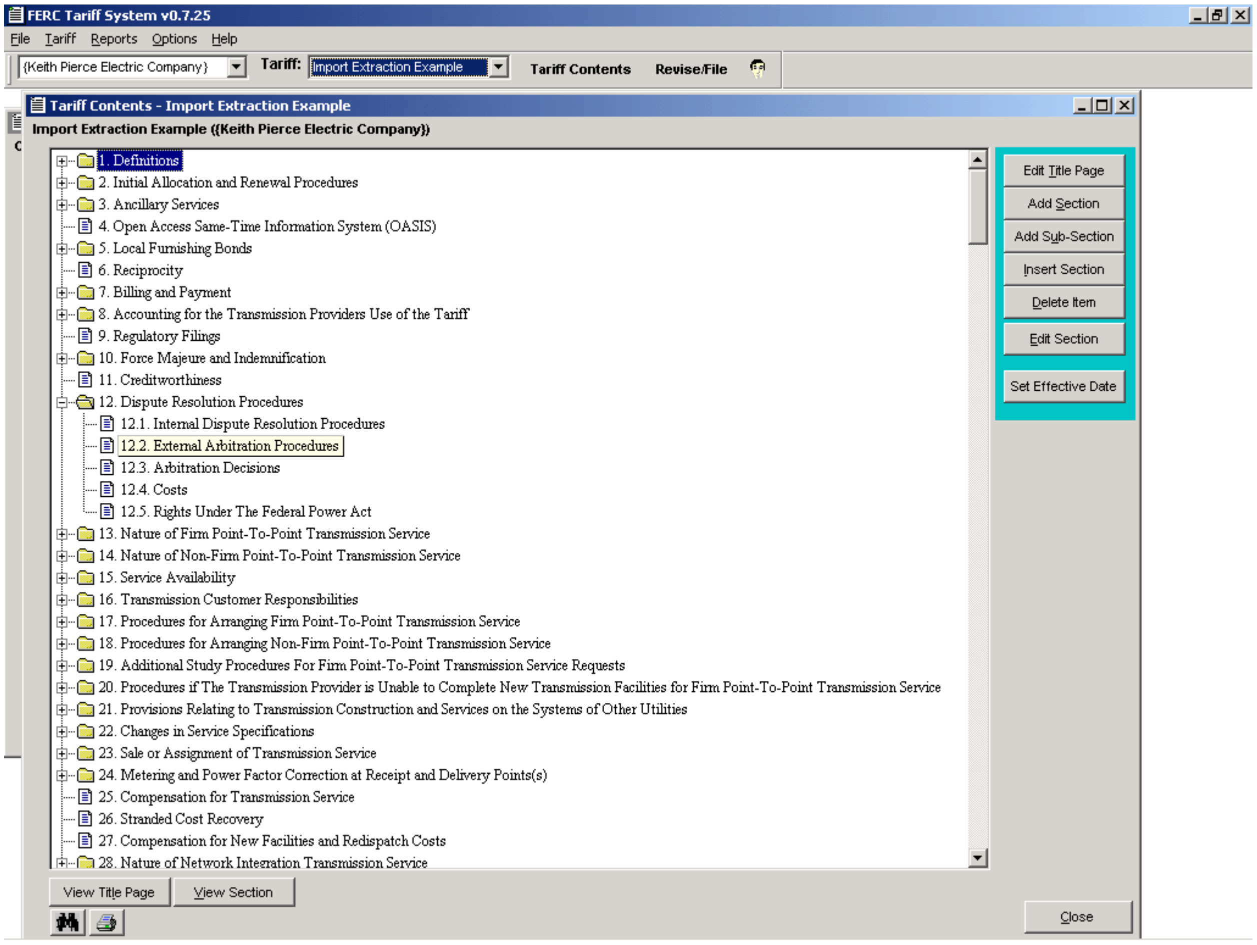


```
1.DOC      132.DOC    166.DOC    2.DOC      46.DOC     8.DOC
10.DOC     133.DOC    167.DOC    20.DOC     47.DOC     80.DOC
100.DOC    134.DOC    168.DOC    200.DOC    48.DOC     81.DOC
101.DOC    135.DOC    169.DOC    201.DOC    49.DOC     82.DOC
102.DOC    136.DOC    17.DOC     202.DOC    5.DOC      83.DOC
103.DOC    137.DOC    170.DOC    203.DOC    50.DOC     84.DOC
104.DOC    138.DOC    171.DOC    204.DOC    51.DOC     85.DOC
105.DOC    139.DOC    172.DOC    205.DOC    52.DOC     86.DOC
106.DOC    14.DOC     173.DOC    206.DOC    53.DOC     87.DOC
107.DOC    140.DOC    174.DOC    207.DOC    54.DOC     88.DOC
108.DOC    141.DOC    175.DOC    21.DOC     55.DOC     89.DOC
109.DOC    142.DOC    176.DOC    22.DOC     56.DOC     9.DOC
11.DOC     143.DOC    177.DOC    23.DOC     57.DOC     90.DOC
110.DOC    144.DOC    178.DOC    24.DOC     58.DOC     91.DOC
111.DOC    145.DOC    179.DOC    25.DOC     59.DOC     92.DOC
112.DOC    146.DOC    18.DOC     26.DOC     6.DOC      93.DOC
113.DOC    147.DOC    180.DOC    27.DOC     60.DOC     94.DOC
114.DOC    148.DOC    181.DOC    28.DOC     61.DOC     95.DOC
115.DOC    149.DOC    182.DO                62.DOC     96.DOC
116.DOC    15.DOC     183.DO                63.DOC     97.DOC
117.DOC    150.DOC    184.DO                64.DOC     98.DOC
118.DOC    151.DOC    185.DOC    31.DOC     65.DOC     99.DOC
119.DOC    152.DOC    186.DOC    32.DOC     66.DOC     Word2eTariff.dbf
12.DOC     153.DOC    187.DOC    33.DOC     67.DOC
120.DOC    154.DOC    188.DOC    34.DOC     68.DOC
121.DOC    155.DOC    189.DOC    35.DOC     69.DOC
122.DOC    156.DOC    19.DOC     36.DOC     7.DOC
123.DOC    157.DOC    190.DOC    37.DOC     70.DOC
124.DOC    158.DOC    191.DOC    38.DOC     71.DOC
125.DOC    159.DOC    192.DOC    39.DOC     72.DOC
126.DOC    16.DOC     193.DOC    4.DOC      73.DOC
127.DOC    160.DOC    194.DOC    40.DOC     74.DOC
128.DOC    161.DOC    195.DOC    41.DOC     75.DOC
129.DOC    162.DOC    196.DOC    42.DOC     76.DOC
13.DOC     163.DOC    197.DOC    43.DOC     77.DOC
130.DOC    164.DOC    198.DOC    44.DOC     78.DOC
131.DOC    165.DOC    199.DOC    45.DOC     79.DOC

Type: Microsoft Word Document
Author: cwnci1
Size: 20.0 KB

208 object(s) (Disk free space: 4.58 GB)    4.24 MB    My Computer
```

The imported tariff from this data set appears as shown below in the eTariff software. Section 12 has been opened to show how the extraction program can find and locate sub-sections:



Import Extraction Example ({Keith Pierce Electric Company})

1. Definitions
2. Initial Allocation and Renewal Procedures
3. Ancillary Services
4. Open Access Same-Time Information System (OASIS)
5. Local Furnishing Bonds
6. Reciprocity
7. Billing and Payment
8. Accounting for the Transmission Providers Use of the Tariff
9. Regulatory Filings
10. Force Majeure and Indemnification
11. Creditworthiness
12. Dispute Resolution Procedures
    12.1. Internal Dispute Resolution Procedures
    12.2. External Arbitration Procedures
    12.3. Arbitration Decisions
    12.4. Costs
    12.5. Rights Under The Federal Power Act
13. Nature of Firm Point-To-Point Transmission Service
14. Nature of Non-Firm Point-To-Point Transmission Service
15. Service Availability
16. Transmission Customer Responsibilities
17. Procedures for Arranging Firm Point-To-Point Transmission Service
18. Procedures for Arranging Non-Firm Point-To-Point Transmission Service
19. Additional Study Procedures For Firm Point-To-Point Transmission Service Requests
20. Procedures if The Transmission Provider is Unable to Complete New Transmission Facilities for Firm Point-To-Point Transmission Service
21. Provisions Relating to Transmission Construction and Services on the Systems of Other Utilities
22. Changes in Service Specifications
23. Sale or Assignment of Transmission Service
24. Metering and Power Factor Correction at Receipt and Delivery Points(s)
25. Compensation for Transmission Service
26. Stranded Cost Recovery
27. Compensation for New Facilities and Redispatch Costs
28. Nature of Network Integration Transmission Service

Appendix

[Note that this document's PDF file was created using the Adobe print add-in for Word. Thus the following code can be copied and pasted as text from this PDF document.]

Example #1: VBA Code for Parsing a Word-formatted tariff into the eTariff database format for automated importing

This example uses VBA and ADODB to pull out each section of the tariff and to put it into the correct format for the eTariff Import Interface guidelines. The test Word document in this case **does not** use numbered paragraphs. It finds each section number by searching the beginning of each paragraph for a numeric digit of a ".". this same logic could be used in a text document scenario.

```
Option Explicit
'Declare the connection object
'NOTE:  Before you can use the connection and recordset objects
'You must set a Reference to the ADODB object library
Dim myconn As ADODB.Connection
Private Sub SearchThru()
'This sub cycles through all the paragraphs in the document,
'picking out the section numbers and section titles, etc.

Dim mypara As Paragraph
Dim mynewfile As Word.Document, myfile As Word.Document
Dim connstr As String, mystr As String
Dim i As Integer, j As Integer
Dim sectnum As String, assocfile As String, parent_sec As String
Dim sect_id As Integer, parent_id As Integer
Dim paratext As String, prevfile As String, secttitle As String, newtext As String
Dim mypath As String

'Set the path
'Everything should reside in this directory
mypath = "D:\ImporteTariff\"

'Build the connection string
connstr = "Provider=vfpoledb;Data Source=D:\ImporteTariff\Word2eTariff.dbf;"

Set myconn = New ADODB.Connection 'set the object
myconn.Open connstr 'open the connection

'NOTE:  The provider connection

'Set id vars
parent_id = 0
sect_id = 1
```

```
For Each mypara In ThisDocument.Paragraphs 'loop thru paragraph collection

  sectnum = "" 'reset vars for each iteration
  assocfile = ""

  'Clean the string
  If mypara.Range.Words.Count > 2 Then
    paratext = CleanString(mypara.Range.Text)
  Else
    'if the string isn't more than 2 chars then move on
    paratext = mypara.Range.Text
  End If

  'Sift thru the paragraph text 1 char at a time
  'searching for a numeric value or a "."
  For i = 1 To Len(paratext)
    If IsNumeric(Mid(paratext, i, 1)) Or Mid(paratext, i, 1) = "." Then

    Else 'no more dots or numbers
      'for paragraph without a section #
      'belongs to previous paragraph
      If i = 1 Then 'no section number, append to previous
        Set myfile = Documents.Open(mypath & prevfile) 'open the previous document
        myfile.Range.Text = myfile.Range.Text & paratext 'put text from existing document
        myfile.Save 'save the changes
        myfile.Close 'close the file
        Exit For 'break out of the For loop since our work is done here
      Else ' section number exists let's write to db
        sectnum = Left(paratext, i - 1) 'assign the section number

        'just in case it doesn't end in a "."
        If Right(sectnum, 1) <> "." Then
          sectnum = sectnum & "."
        End If

        'Find parent section
        If Len(sectnum) > 2 Then
          For j = Len(sectnum) - 1 To 1 Step -1
            If Mid$(sectnum, j, 1) = "." Then
              'found second "."
              parent_sec = Left(sectnum, j)
              'Look up the parent id
              parent_id = LookupParent(parent_sec)
              Exit For
            End If
          Next j
        Else
          'must be a top level
          parent_id = 0
        End If
        'Get title if it is present
        'All titles are bold in our test tariff

        If mypara.Range.Bold = True Or mypara.Range.Bold = "9999999" Then
          'secttitle = Right(mypara.Range.Text, Len(mypara.Range.Text) - i)
          secttitle = CleanString(Right(mypara.Range.Text, Len(mypara.Range.Text) - i))
        End If

        Set mynewfile = Documents.Add 'Create new document
        mynewfile.Range.Text = paratext 'Put text in new document

        mynewfile.SaveAs FileName:=mypath & sect_id & ".DOC" 'save file

        mynewfile.Close 'close file
        prevfile = sect_id & ".DOC"

        assocfile = sect_id & ".DOC" 'set filename of doc based on the section number

        'Write into database
        mystr = "INSERT INTO Word2eTariff (assoc_file, sect_id, parent_id, sect_num, sect_title) VALUES ('" & assocfile & "', " & sect_id & ", " & parent_id & ", '" & sectnum & "', '" & secttitle & "')"
        'MsgBox mystr
        myconn.Execute mystr

        sect_id = sect_id + 1 'Increment section ID


        Exit For
      End If

    End If

  Next i

Next mypara

End Sub
Private Function LookupParent(parent_sec As String) As Integer
'This function looks up the section id of a parent section
'The return value of this function will be used for the parent_id field of a new row

Dim myrst As ADODB.Recordset 'create a recordset to store rows for look up
'execute the SELECT statement, all rows will be returned in the recorset object
Set myrst = myconn.Execute("SELECT sect_id, sect_num FROM Word2eTariff")

'Loop thru the recordset and match parent_sec with sect_num
'and return the sect_id when found.
Do While Not myrst.EOF
  If Trim(myrst!sect_num) = Trim(parent_sec) Then
    LookupParent = myrst!sect_id 'return the sect_id
    Set myrst = Nothing
    Exit Function
  End If
  myrst.MoveNext 'move to the next record
Loop

MsgBox "Error!  Could not find section id to reference the parent id."

Set myrst = Nothing 'destroy objects = a good practice
LookupParent = 0 'return 0 if an error occurs; 0 = a top level section with no parent

End Function
Private Function CleanString(mystr As String) As String

'This function strikes all the chars that are non alpha and non numeric
'from the left and the right of the string that is the current paragraph.
'It returns a clean string.
'Otherwise how can you test to see if the beginning chars are
'section numbers if the paragraph starts with an Indent or Tab char or whatever.

'Dim i As Integer 'char counter
Dim done As Boolean 'flag variable

mystr = Trim(mystr) 'Trim spaces from left and right

'Clean Left side of the string
Do Until Asc(Left(mystr, 1)) > 33 And Asc(Left(mystr, 1)) < 126
  mystr = Right(mystr, Len(mystr) - 1)
Loop

'Clean Right side of the string
Do Until Asc(Right(mystr, 1)) > 33 And Asc(Right(mystr, 1)) < 126
  mystr = Left(mystr, Len(mystr) - 1)
Loop


'Return the cleaned string
CleanString = mystr


End Function

Private Sub ViewDB()
'This sub was used to quickly view the records that have been added to the database
'Viewing the table in FoxPro is ideal.  MS Access can also be used to view the records
'but only after converting the FoxPro table to Access (.MDB)

Dim myrst As ADODB.Recordset
Dim myconn As ADODB.Connection
Dim connstr As String
```

```
Set myconn = New ADODB.Connection 'create connection object
connstr = "Provider=vfpoledb;Data Source=D:\ImporteTariff\Word2eTariff.dbf;"
myconn.Open connstr 'open the connection

'Select all records for viewing
Set myrst = myconn.Execute("Select * FROM Word2eTariff")

'Loop thru and display each record using a message box.
Do While Not myrst.EOF
  MsgBox myrst!sect_id & " -- " & myrst!parent_id & " -- " & myrst!sect_num & " -- " & myrst!assoc_file
  myrst.MoveNext
Loop


End Sub
```

Example #2:
VBA Code for Parsing a Word-formatted tariff w/numbered paragraphs into the eTariff database format for automated importing

This example uses VBA and ADODB to pull out each section of the tariff and to put it into the correct format for the eTariff Import Interface guidelines. The test Word document in this case **includes** numbered paragraphs. It finds each section number by searching the beginning of each paragraph for a “listed” number. Any paragraph without a number gets appended to the previous section.

```
Option Explicit
'Declare the connection object
'NOTE:  Before you can use the connection and recordset objects
'You must set a Reference to the ADODB object library
Dim myconn As ADODB.Connection

Private Sub SearchThru()
'This sub cycles through all the paragraphs in the document,
'picking out the section numbers and section titles, etc.

Dim mypara As Paragraph
Dim mynewfile As Word.Document, myfile As Word.Document
Dim connstr As String, mystr As String
Dim i As Integer, j As Integer, dots As Integer
Dim sectnum As String, assocfile As String, parent_sec As String
Dim sect_id As Integer, parent_id As Integer
Dim prevfile As String, secttitle As String, newtext As String
Dim mypath As String

'Set the path
'Everything should reside in this directory
mypath = "K:\ImporteTariff\"

'Build the connection string
```

```
connstr = "Provider=vfpoledb;Data Source=K:\ImporteTariff\Word2eTariff.dbf;"

Set myconn = New ADODB.Connection 'set the object
myconn.Open connstr 'open the connection

'Set id vars
parent_id = 0
sect_id = 1


For Each mypara In ThisDocument.Paragraphs 'loop thru paragraphs collection
  sectnum = "" 'reset vars for each iteration
  assocfile = ""

  'Test for ListParagraph
  If mypara.Range.ListFormat.ListType > 0 Then 'it's a ListParagraph of some sort
    sectnum = mypara.Range.ListFormat.ListString

    'just in case it doesn't end in a "."
    If Right(sectnum, 1) <> "." Then
      sectnum = sectnum & "."
    End If


    'if 1 dot only then it's a parent
    dots = 0 'reset for each iteration
        'count the dots
        For j = 1 To Len(sectnum)
          If Mid$(sectnum, j, 1) = "." Then
            dots = dots + 1
          End If
        Next j

    'Find parent section
        If dots > 1 Then
          For j = Len(sectnum) - 1 To 1 Step -1
            If Mid$(sectnum, j, 1) = "." Then
              'found second "."
              parent_sec = Left(sectnum, j)
              'Look up the parent id
              parent_id = LookupParent(parent_sec)
              Exit For
            End If
          Next j
        Else
          'must be a top level
          parent_id = 0
        End If

    If mypara.Range.Bold = True Or mypara.Range.Bold = "9999999" Then '9999999 = mixed
```

```
      secttitle = CleanString(mypara.Range.Text)
    End If

    Set mynewfile = Documents.Add 'Create new document
    'mynewfile.Range.Text = CleanString(mypara.Range.Text) 'Clean String & put text in new document

    mynewfile.SaveAs FileName:=mypath & sect_id & ".DOC" 'save file

    mynewfile.Close 'close file
    prevfile = sect_id & ".DOC"

    assocfile = sect_id & ".DOC" 'set filename of doc based on the section number

    'Write into database
    mystr = "INSERT INTO Word2eTariff (assoc_file, sect_id, parent_id, sect_num, sect_title) VALUES ('" & assocfile & "', " & sect_id & ", " & parent_id & ", '" & sectnum & "', '" & secttitle & "')"
    myconn.Execute mystr

    sect_id = sect_id + 1 'Increment section ID

  Else 'append to previous doc
      Set myfile = Documents.Open(mypath & prevfile) 'open the previous document

      myfile.Range.Text = myfile.Range.Text & mypara.Range.Text 'put text from existing document
      myfile.Save 'save the changes
      myfile.Close 'close the file
  End If
Next mypara

End Sub
Private Function LookupParent(parent_sec As String) As Integer
'This function looks up the section id of a parent section
'The return value of this function will be used for the parent_id field of a new row

Dim myrst As ADODB.Recordset 'create a recordset to store rows for look up
'execute the SELECT statement, all rows will be returned in the recorset object
Set myrst = myconn.Execute("SELECT sect_id, sect_num FROM Word2eTariff")

'Loop thru the recordset and match parent_sec with sect_num
'and return the sect_id when found.
Do While Not myrst.EOF
  If Trim(myrst!sect_num) = Trim(parent_sec) Then
    LookupParent = myrst!sect_id 'return the sect_id
    Set myrst = Nothing
    Exit Function
  End If
  myrst.MoveNext 'move to the next record
Loop

MsgBox "Error!  Could not find section id to reference the parent id."
```

```
Set myrst = Nothing 'destroy objects = a good practice
LookupParent = 0 'return 0 if an error occurs; 0 = a top level section with no parent

End Function

Private Function CleanString(mystr As String) As String

'This function strikes all the chars that are non alpha and non numeric
'from the left and the right of the string that is the current paragraph.
'It returns a clean string.
'Otherwise how can you test to see if the beginning chars are
'section numbers if the paragraph starts with an Indent or Tab char or whatever.

mystr = Trim(mystr) 'Trim spaces from left and right

'Clean Left side of the string
Do Until Asc(Left(mystr, 1)) > 33 And Asc(Left(mystr, 1)) < 126
  mystr = Right(mystr, Len(mystr) - 1)
Loop

'Clean Right side of the string
Do Until Asc(Right(mystr, 1)) > 33 And Asc(Right(mystr, 1)) < 126
  mystr = Left(mystr, Len(mystr) - 1)
Loop

'Return the cleaned string
CleanString = mystr

End Function
Private Sub makebold()

'This subroutine steps thru all the "listed' paragraphs in a word document
'and makes it bold.  This is how I bolded all the titles in the document
Dim numpar As Paragraph

For Each numpar In ThisDocument.ListParagraphs
    numpar.Range.Bold = True
Next numpar

End Sub
Private Sub ViewDB()
'This sub was used to quickly view the records that have been added to the database
'Viewing the table in FoxPro is ideal.  MS Access can also be used to view the records
'but only after converting the FoxPro table to Access (.MDB)

Dim myrst As ADODB.Recordset
Dim myconn As ADODB.Connection
Dim connstr As String

Set myconn = New ADODB.Connection 'create connection object
```

```
connstr = "Provider=vfpoledb;Data Source=K:\ImporteTariff\Word2eTariff.dbf;"
myconn.Open connstr 'open the connection

'Select all records for viewing
Set myrst = myconn.Execute("Select * FROM Word2eTariff")

'Loop thru and display each record using a message box.
Do While Not myrst.EOF
  MsgBox myrst!sect_id & " -- " & myrst!parent_id & " -- " & myrst!sect_num & " -- " & myrst!assoc_file
  myrst.MoveNext
Loop


End Sub
```

Example #3:
VBA Code for Parsing a Excel-formatted tariff into the eTariff database format for automated importing

This example uses VBA and ADODB to pull out each section of the tariff and to put it into the correct format for the eTariff Import Interface guidelines. It finds each section number by searching the beginning of each paragraph for a “listed” number. Any paragraph without a number gets appended to the previous section.

```
Option Explicit
Dim myconn As ADODB.Connection
Dim wd As Word.Application
Private Sub GetData()
Dim mynewfile As Word.Document

Dim i As Integer, j As Integer
Dim connstr As String, mystr As String, mypath As String, lastsection As String
Dim sectnum As String, assocfile As String
Dim prevfile As String, secttitle As String, newtext As String
Dim sect_id As Integer

'Set path
mypath = "K:\ImporteTariff\Test4\"

'create connection string
connstr = "Provider=vfpoledb;Data Source=K:\ImporteTariff\Test4\word2etariff.dbf;"

Set myconn = New ADODB.Connection 'create the connection object
myconn.Open connstr 'open the table

Set wd = New Word.Application
sect_id = 1
i = 10 'make it start at 11

'get data
```

```
Do
  i = i + 1 'row counter
  sectnum = ThisWorkbook.Worksheets("PAGE1").Cells(i, 1)

  If sectnum = "END" Then 'break out at the end
    Exit Do
  End If

  If sectnum <> "" Then
    Set mynewfile = wd.Documents.Add
    'sectnum = mystr

    If Right(sectnum, 1) <> "." Then
      sectnum = sectnum & "."
    End If

    For j = 2 To 15 'check accross the sheet for any data
      If ThisWorkbook.Worksheets("PAGE1").Cells(i, j) <> "" Then
        newtext = newtext & " " & ThisWorkbook.Worksheets("PAGE1").Cells(i, j)
      End If
    Next j

    mynewfile.Range.Text = newtext
    mynewfile.SaveAs Filename:=mypath & sect_id & ".doc"
    assocfile = sect_id & ".doc"
    prevfile = sect_id & ".doc"
    mynewfile.Close

    'Write into database
    mystr = "INSERT INTO Word2eTariff (assoc_file, sect_id, parent_id, sect_num, sect_title) VALUES ('" & assocfile & "', " & sect_id & ", 0, '" & sectnum & "', '<no title>')"
    myconn.Execute mystr

    sect_id = sect_id + 1

  Else 'append previous section

      For j = 2 To 15 'check accross the sheet for any data
        If ThisWorkbook.Worksheets("PAGE1").Cells(i, j) <> "" Then
          newtext = newtext & " " & ThisWorkbook.Worksheets("PAGE1").Cells(i, j)
        End If
      Next j

      If newtext <> "" Then
        Set mynewfile = wd.Documents.Open(mypath & prevfile)
        mynewfile.Range.Text = mynewfile.Range.Text & newtext
        mynewfile.Save
        mynewfile.Close
      End If

  End If
```

```
newtext = ""
mystr = ""

Loop

Call PageTwo(sect_id)

myconn.Close
Set mynewfile = Nothing
Set myconn = Nothing
Set wd = Nothing

End

End Sub
Private Sub PageTwo(lastsect_id As Integer)
  Dim i As Integer 'row counter
  Dim mystr As String, newsect As String, assocfile As String
  Dim mypath As String, secttitle As String, lastsection As String
  Dim mynewfile As Word.Document

  secttitle = ThisWorkbook.Worksheets("PAGE 2").Cells(3, 2)
  lastsection = "11."
  mypath = "K:\ImporteTariff\Test4\"

  'build string of data
  Do
    i = i + 1
    If ThisWorkbook.Worksheets("PAGE 2").Cells(i, 6) = "END" Then
      Exit Do
    ElseIf ThisWorkbook.Worksheets("PAGE 2").Cells(i, 6) <> "" Then
      mystr = mystr & " " & ThisWorkbook.Worksheets("PAGE 2").Cells(i, 6)
    End If
  Loop

  Set mynewfile = wd.Documents.Add
  mynewfile.Range.Text = mystr
  mynewfile.SaveAs Filename:=mypath & lastsect_id & ".doc"
  assocfile = lastsect_id & ".doc"
  mynewfile.Close

  'Write into database
  mystr = "INSERT INTO Word2eTariff (assoc_file, sect_id, parent_id, sect_num, sect_title) VALUES ('" & assocfile & "', " & lastsect_id & ", 0, '" & lastsection & "', '" & secttitle & "')"
  myconn.Execute mystr

  Set mynewfile = Nothing
  'Set wd = Nothing
  'myconn.Close
```

```
End Sub
Private Function CleanString(mystr As String) As String

'This function strikes all the chars that are non alpha and non numeric
'from the left and the right of the string that is the current paragraph.
'It returns a clean string.
'Otherwise how can you test to see if the beginning chars are
'section numbers if the paragraph starts with an Indent or Tab char or whatever.

mystr = Trim(mystr) 'Trim spaces from left and right

'Clean Left side of the string
Do Until Asc(Left(mystr, 1)) > 33 And Asc(Left(mystr, 1)) < 126
  mystr = Right(mystr, Len(mystr) - 1)
Loop

'Clean Right side of the string
Do Until Asc(Right(mystr, 1)) > 33 And Asc(Right(mystr, 1)) < 126
  mystr = Left(mystr, Len(mystr) - 1)
Loop

'Return the cleaned string
CleanString = mystr

End Function
Private Sub ViewDB()
'This sub was used to quickly view the records that have been added to the database
'Viewing the table in FoxPro is ideal.  MS Access can also be used to view the records
'but only after converting the FoxPro table to Access (.MDB)

Dim myrst As ADODB.Recordset
Dim myconn As ADODB.Connection
Dim connstr As String

Set myconn = New ADODB.Connection 'create connection object
connstr = "Provider=vfpoledb;Data Source=K:\ImporteTariff\Test4\Word2eTariff.dbf;"
myconn.Open connstr 'open the connection

'Select all records for viewing
Set myrst = myconn.Execute("delete FROM Word2eTariff")
End

'Loop thru and display each record using a message box.
Do While Not myrst.EOF
  MsgBox myrst!sect_id & " -- " & myrst!parent_id & " -- " & myrst!sect_num & " -- " & myrst!assoc_file
  myrst.MoveNext
Loop


End Sub
```